Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2009, Article ID 250794, 17 pages
doi:10.1155/2009/250794

Research Article

Hardware Realization of Generalized Time-Frequency Distribution with Complex-Lag Argument

Nikola Žarić, Irena Orović, and Srdjan Stanković

Faculty of Electrical Engineering, University of Montenegro, 20000 Podgorica, Montenegro

Correspondence should be addressed to Nikola Žarić, zaric@ac.me

Received 13 July 2009; Accepted 2 December 2009

Recommended by Patrick Oonincx

A hardware implementation of the *N*th order complex-lag time-frequency distribution is proposed. The considered distribution provides an arbitrary high concentration for multicomponent signals with fast varying instantaneous frequency. Although the distribution form is quite complex, the proposed realization is very efficient and provides high-speed real-time processing. Further, it allows avoiding miscalculation errors that may appear in the numerical calculation of signal with complex-lag argument. The results of FPGAs (Field Programmable Gate Arrays) implementation are presented as well.

1. Introduction

Time-frequency analyses have been used in many applications with nonstationary signals, such as speech, radar, seismic, biomedical, and communication signals, and so forth. Various time-frequency distributions have been introduced to provide optimal representation in the time-frequency domain. For instance, an ideal representation for linear frequency modulated signals is obtained by using the quadratic distributions [1–4]. However, they cannot provide satisfying distribution concentration for signals with higher nonstationarity, since the inner interferences appear. In order to improve concentration, various higher-order distributions are used: L-Wigner distribution [5], Polynomial distributions [6, 7], as well as the distributions with complex-lag argument [8–12]. Recently, the *N*th order general form of complex-lag time-frequency distribution for multicomponent signals has been proposed [13]. This form can provide a cross-terms free representation with an arbitrary high concentration even for signals with significant phase variations within a few samples. Although it produces very efficient results, MATLAB simulation of the *N*th order complex-lag distribution requires significant computational time, inappropriate for real-time processing. Also, the software simulation has a significant latency and low throughput rate. These drawbacks of the software simulation could be solved by a suitable hardware realization. Consequently, the favorable properties of the complex-lag distribution could be available in various real-time applications.

A flexible architecture of an arbitrary (*N*th) order complex-lag distribution for multicomponent signals is proposed in this paper. Although it seems that the *N*th order complex-lag distribution is difficult for realization, an efficient parallel configuration is provided, suitable for VLSI implementation that allows high-speed real-time processing. The serial realization that reduces number of components is considered as well. The proposed hardware overcomes the errors in the numerical calculations caused by the limited MATLAB software precision, which is an additional advantage. Namely, this problem appears in the calculation of analytic extension of the signal with complex-lag argument [9, 13].

The proposed system consists of two main parts: realization of the S-method (proposed in [14]) and realization of the concentration function (system of order *N*-2). The realization of the concentration function is performed throughout several subsystems providing calculation of the signal with complex-lag argument, concentration function calculation, and its time-frequency domain version. The parallel architecture is implemented on the FPGAs chips in order to verify the results. A simple solution for some intermediate functions (based on logarithmic and sine and

cosine functions) that do not exist as standard components, is given as well.

The paper is organized as follows. The review of generalized time-frequency distribution with complex-lag argument is given in Section 2. Section 3 presents the parallel hardware realization of generalized time-frequency distribution with complex-lag argument, while in Section 4 the serial realization is proposed. Section 5 presents the analysis and comparison of the proposed system. The FPGA implementation and simulation results are given in Section 6. Concluding remarks are given in Section 7. The realizations of intermediate calculations are given in the appendix.

2. Theoretical Background

A general discrete form of the Nth order time-frequency distribution with complex-lag argument can be written as [13]

$$GCD_N(n,k) = \sum_{m=-N_s/2}^{N_s/2-1} x(n+m)\overline{x}(n-m)c(n,m)e^{-j(2\pi/N_s)Nmk}, \tag{1}$$

where $\overline{(\cdot)}$ denotes complex conjugation, n and k are discrete time and frequency variables, respectively, N_s is the number of samples, while $c(n,m)$ represents the concentration function:

$$c(n,m) = \prod_{p=1}^{N/2-1} x^{\overline{w}_{N,p}}\left(n + m\frac{w_{N,p}}{N}\right)x^{-\overline{w}_{N,p}}\left(n - m\frac{w_{N,p}}{N}\right), \tag{2}$$

where $w_{N,p} = e^{j2\pi p/N} = w_{r_p} + jw_{i_p}$. Observe that the $GCD_N(n,k)$ can be expressed as

$$\begin{aligned} GCD_N(n,k) &= \frac{N}{2}WD\left(n,\frac{N}{2}k\right) *_k \mathrm{FT}_m(c(n,m)) \\ &= \frac{N}{2}WD\left(n,\frac{N}{2}k\right) *_k C(n,k). \end{aligned} \tag{3}$$

The convolution in frequency domain is denoted by $*_k$, while FT_m denotes the Fourier transform with respect to variable m. The concentration function (of order N-2) acts as a correction term that can arbitrarily improve the concentration of the Wigner distribution (WD) by increasing the order N. In order to provide a suitable representation for multicomponent signals, the Nth order complex-lag distribution has been modified [13]. The modifications are introduced in the calculation of the concentration function (for the qth signal component): $c(n,m)_q = c_r(n,m)_q \cdot c_i(n,m)_q$, where

$$\begin{aligned} c_r(n,m)_q &= \prod_{p=1}^{N/2-1} c_{r_p}(n,m)_q \\ &= \prod_{p=1}^{N/2-1} e^{jw_{r_p}angle(x_{a_p}(n+m(w_{N,p}/N))_q \cdot \overline{x}_{a_p}(n-m(w_{N,p}/N))_q)}, \\ c_i(n,m)_q &= \prod_{p=1}^{N/2-1} c_{i_p}(n,m)_q \\ &= \prod_{p=1}^{N/2-1} e^{-jw_{i_p}\ln|x_{a_p}(n+m(w_{N,p}/N))_q \cdot \overline{x}_{a_p}(n-m(w_{N,p}/N))_q|}. \end{aligned} \tag{4}$$

More details could be found in [13]. Calculation of signal with complex-lag argument $x_{a_p}(n \pm m(w_{N,p}/N))_q$ is considered later in this section.

The time-frequency representations of the resulting functions $c_r(n,m)$ and $c_i(n,m)$ (for all signal components) are obtained as

$$\begin{aligned} C_r(n,k) &= \mathrm{FT}_m\left\{\sum_{q=1}^{Q} c_r(n,m)_q\right\}, \\ C_i(n,k) &= \mathrm{FT}_m\left\{\sum_{q=1}^{Q} c_i(n,m)_q\right\}, \end{aligned} \tag{5}$$

where Q is the number of signal components.

The final form of the concentration function in the time-frequency domain is the convolution of $C_r(n,k)$ and $C_i(n,k)$:

$$C(n,k) = \sum_{i=-L_d}^{L_d} P(i)C_r(n,k+i)C_i(n,k-i). \tag{6}$$

Note that $P(i)$ is a frequency domain window of the size $2L_d+1$. The cross-terms free representation is obtained if the size of window is less than the minimal distance between the autoterms.

Finally, a general form of modified complex-lag time-frequency distribution is defined as

$$\mathrm{MGCD}_N(n,k) = \sum_{i=-L_d}^{L_d} P(i)\mathrm{SM}(n,k+i)C(n,k-i), \tag{7}$$

where, instead of the WD, the S-method (cross-terms free WD), $\mathrm{SM}(n,k) = \sum_{i=-L_d}^{L_d} P(i)\mathrm{STFT}(n,k+i)\overline{\mathrm{STFT}}(n,k-i)$, is used. The acronym STFT is used for the short time Fourier transform.

Calculation of signal with complex-lag argument. In order to calculate the signal with complex-lag argument, the signal components should be first separated by using the STFT

[9]. Then the qth component of the signal with complex-lag argument is calculated by using the analytic extension as follows [13]:

$$x_{a_p}\left(n \pm m\frac{w_{N,p}}{N}\right)_q = \sum_{k=k_q-W_q}^{k_q+W_q} \mathrm{STFT}\left(n, k + k_q(n)\right) \times e^{j2\pi(n\pm m(w_{N,p}/N))k/N_s}. \tag{8}$$

It is assumed that the qth signal component is within the region $[k_q(n) - W_q, k_q(n) + W_q]$ where $k_q(n) = \arg\{\max_k \mathrm{STFT}(n,k)\}$ is the position of its maximum. The parameter W_q is used to define the width $(2W_q + 1)$ of the q-th signal component in the time-frequency plane. The cross-terms will be avoided if $2W_q + 1$ is smaller than the distance between autoterms. The width $2W_q + 1$ could be adjusted for each signal component [14].

Note that the real part of exponential function $\exp(jmkw_{N,p}/N)$, for large values of mk, can exceed the computer (software) precision range, and it may cause errors in the numerical realization.

3. Parallel Architecture for Implementation of the Nth Order Complex-Lag Time-Frequency Distribution

A system for implementation of the Nth order complex-lag time-frequency distribution for multicomponent signals is proposed in this section. The block scheme is given in Figure 1. The starting block is the calculation of the STFT that is used to obtain the S-method ($SM(n,k)$). According to (8), the STFT is also used to obtain the signal with complex-lag argument for the computation of $C(n,k)$ (by using (4), (5), and (6)). The S-method is obtained at the output of the SM block, while the $C(n,k)$ is obtained at the output of the BLOCK 4 in Figure 1. The complex-lag distribution ($\mathrm{MGCD}(n,k)$) is produced at the output of the BLOCK 5, as a convolution of the $\mathrm{SM}(n,k)$ and $C(n,k)$.

3.1. Hardware Solutions for the STFT and the SM. The architectures for the STFT and the SM implementation have been proposed in [14]. They are shown in Figures 2(a) and 2(b), respectively. Namely, by using the rectangular window, the STFT is realized as [15, 16]

$$\begin{aligned}\mathrm{STFT}_{\mathrm{Re}}(n,k) &= (-1)^k\left[x\left(n+\frac{N_s}{2}\right) - x\left(n-\frac{N_s}{2}\right)\right] \\ &\quad + c(k)\mathrm{STFT}_{\mathrm{Re}}(n-1,k) \\ &\quad - s(k)\mathrm{STFT}_{\mathrm{Im}}(n-1,k), \\ \mathrm{STFT}_{\mathrm{Im}}(n,k) &= c(k)\mathrm{STFT}_{\mathrm{Im}}(n-1,k) \\ &\quad + s(k)\mathrm{STFT}_{\mathrm{Re}}(n-1,k),\end{aligned} \tag{9}$$

where, for a given k, the quantities $c(k) = \cos(2\pi k/N_s)$ and $s(k) = \sin(2\pi k/N_s)$ are constants. The $\mathrm{STFT}(n,k)$ represents the input for the SM realization. Thus, the SM is obtained according to the form presented in [14]:

$$\begin{aligned}\mathrm{SM}(n,k) &= |\mathrm{STFT}(n,k)|^2 \\ &\quad + 2\sum_{i=1}^{L_d} STFT_{\mathrm{Re}}(n,k+i)\mathrm{STFT}_{\mathrm{Re}}(n,k-i) \\ &\quad + 2\sum_{i=1}^{L_d} STFT_{\mathrm{Im}}(n,k+i)\mathrm{STFT}_{\mathrm{Im}}(n,k-i).\end{aligned} \tag{10}$$

3.2. Hardware Solution for the Concentration Function in Time-Frequency Domain $C(n,k)$. The parallel architecture for concentration function in the time-frequency domain is realized through several blocks in Figure 1: STFT block, BLOCK 1, BLOCK 2, BLOCK 3, and BLOCK 4. The outputs of the STFT block used in the calculation of the concentration function are $\mathrm{STFT}c(n,k) = (N/2)\mathrm{STFT}(n,(N/2)k)$. The separation of signal components is performed within the BLOCK 1, while the BLOCK 2 is used for $c_r(n,m)$ and $c_i(n,m)$ calculation. The Fourier transforms of these functions ($C_r(n,k)$ and $C_i(n,k)$) are performed in BLOCK 3. The final form of the concentration function $C(n,k)$ (in the time-frequency domain) is obtained at the output of BLOCK 4.

In the sequel, each block will be analyzed and presented separately.

3.2.1. BLOCK 1: Separation of Signal Components. The regions that contain signal components are separated within BLOCK 1, based on the outputs of the STFT block. In this sense, it is necessary first to allocate components of $|\mathrm{STFT}_c(n,k)|$ that are higher than the reference value R. For instance, if the signal amplitudes are normalized, the STFT values are in the range $[-1,1]$, and the reference value should be set to $R = 1/\lambda$, where λ is a scaling constant and $\lambda \geq 1$ holds. Otherwise, if the signal range is unknown, the reference value is defined as a portion of the STFT's maximum at a given instant n, $R = \max_k|\mathrm{STFT}_c(n,k)|/\lambda$. The regions of $\mathrm{STFT}_c(n,k)$ that are higher than R contain the signal components. Each of them is further processed to find the position of its maximal component denoted by k_q, $q = 1,\ldots,Q$, where Q is the number of allocated regions, that is, the number of signal components. The outputs $STFT_c(n,k)$ for $k \in [k_q - W_q, k_q + W_q]$ are passed to the inputs of BLOCK 2.

3.2.2. BLOCK 2: Realization of the Concentration Functions $c_r(n,m)$ and $c_i(n,m)$. Hardware solutions for $x_{a_p}(n+w_{N,p}m/N)_q$ and $x_{a_p}(n-w_{N,p}m/N)_q$ represent an initial step in the realization of BLOCK 2. They are shown in Figures 3(a) and 3(b), respectively. For the pth term in (4) and the qth signal component, the two channels (real and imaginary part) of the signal with complex-lag argument

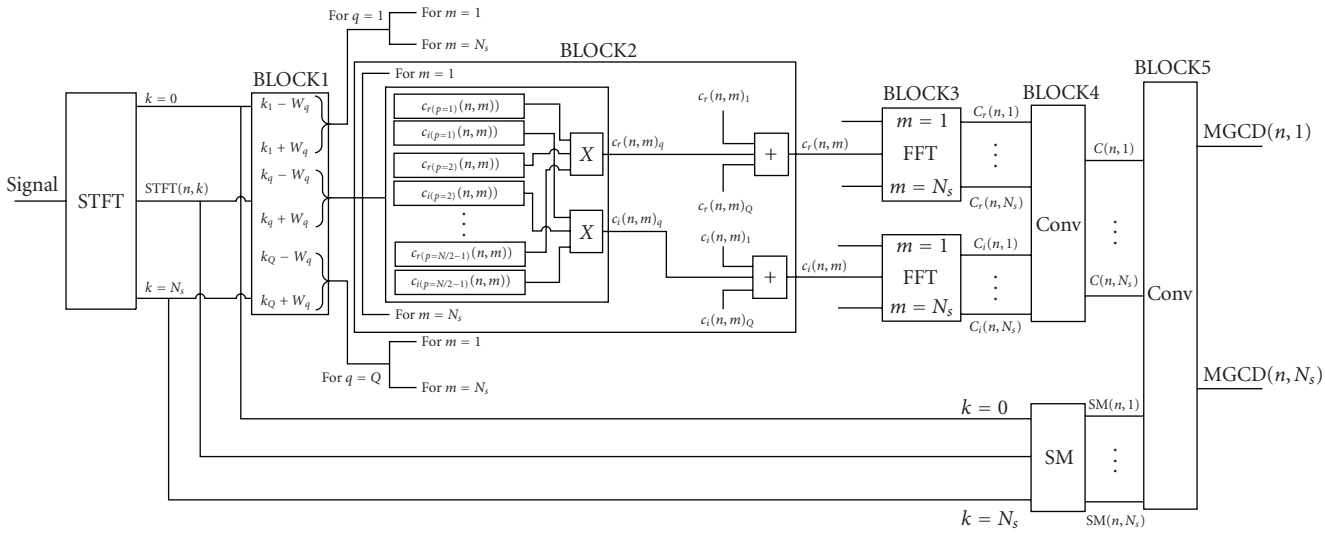


Figure 1: Parallel architecture for realization of generalized complex-lag distribution for multicomponent signals.

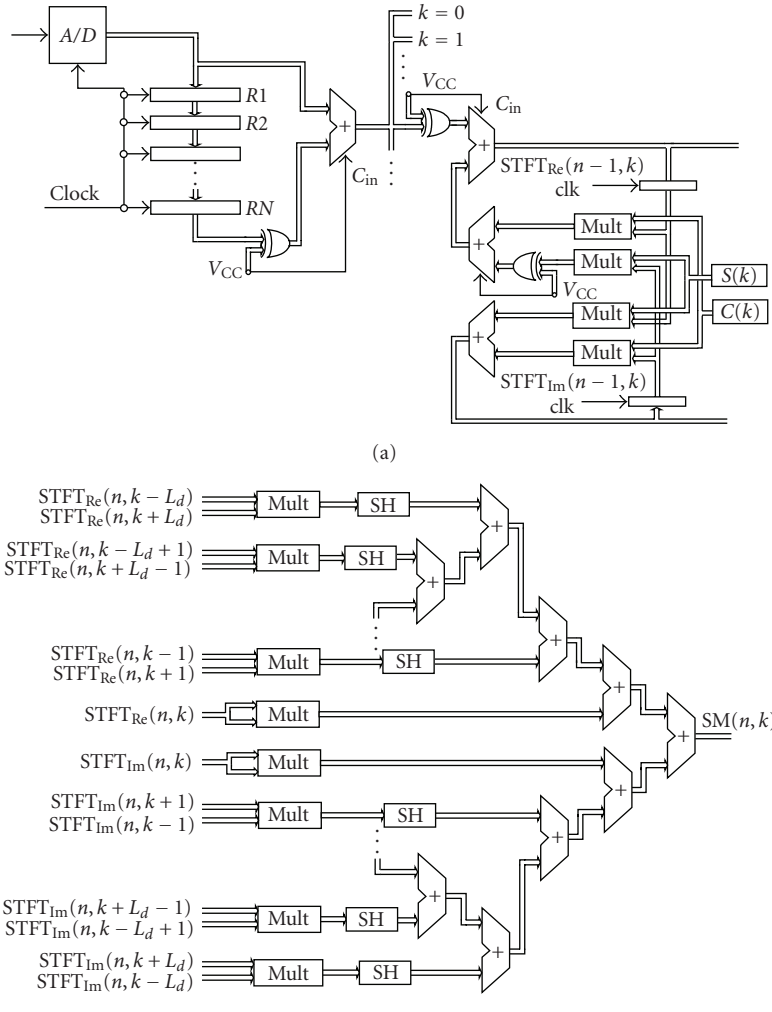
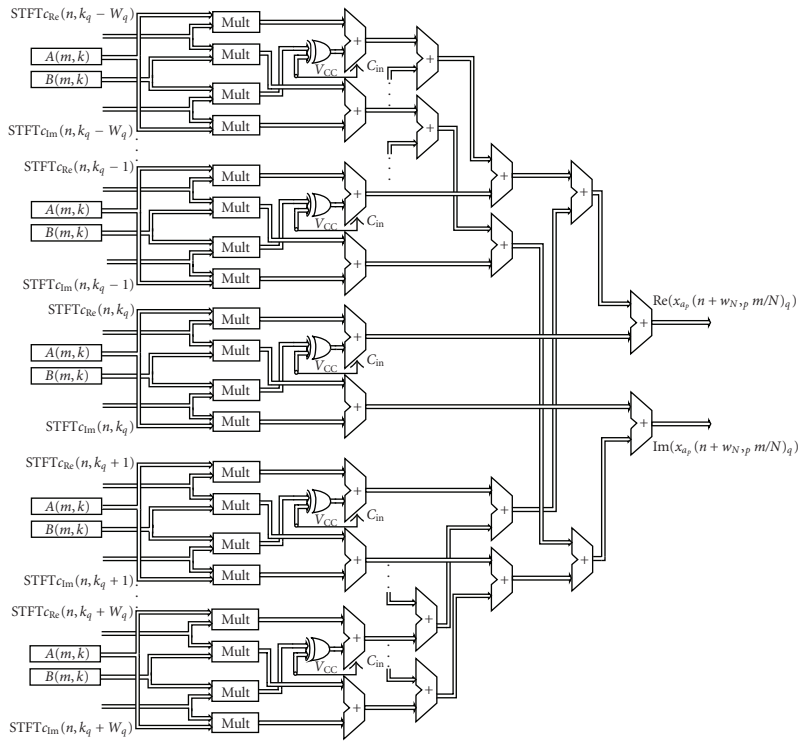
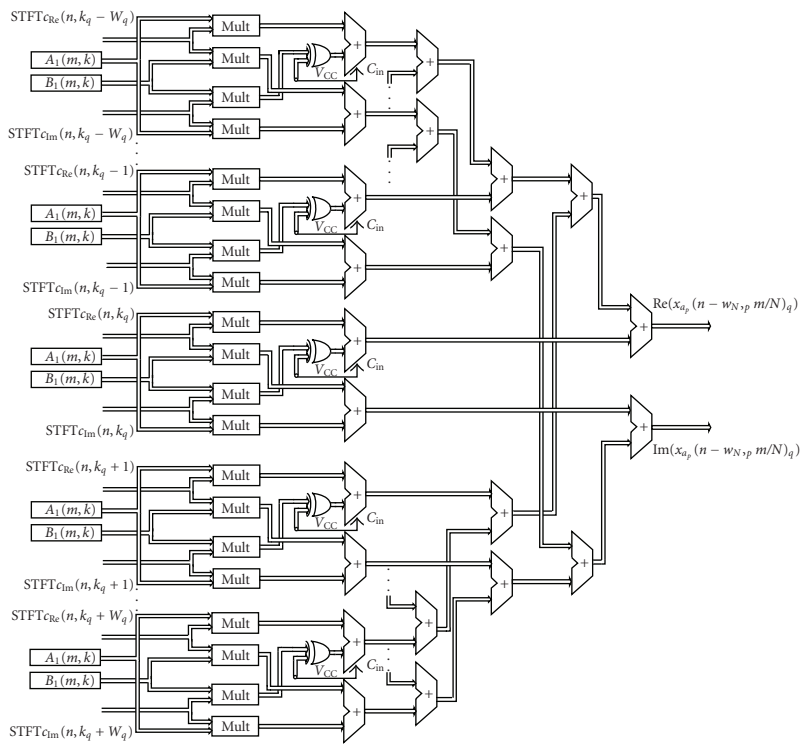


Figure 2: Architecture for realization of (a) STFT and (b) S-method.



(a)



(b)

Figure 3: Architecture for producing of signal with complex-lag argument: (a) $x_{a_p(n+w_{N,p}m/N)_q}$, (b) $x_{a_p(n-w_{N,p}m/N)_q}$.

$x_{a_p}(n + w_{N,p}m/N)_q$ (Figure 3(a)) are obtained according to (8):

$$\begin{aligned}
&\mathrm{Re}\left\{x_{a_p}\left(n + w_{N,p}m/N\right)_q\right\} \\
&= \sum_{k=k_q-W_q}^{k+W_q} \left(A(m,k)\mathrm{STFT}c_{\mathrm{Re}}(n,k) - B(m,k)\mathrm{STFT}c_{\mathrm{Im}}(n,k)\right), \\
&\mathrm{Im}\left\{x_{a_p}\left(n + w_{N,p}m/N\right)_q\right\} \\
&= \sum_{k=k_q-W_q}^{k_q+W_q} \left(A(m,k)\mathrm{STFT}c_{\mathrm{Im}}(n,k) + B(m,k)\mathrm{STFT}c_{\mathrm{Re}}(n,k)\right).
\end{aligned} \tag{11}$$

The constants $A(m,k)$ and $B(m,k)$ are defined as

$$\begin{aligned}
A(m,k) &= (-1)^k \cos\frac{w_{r_p}2\pi km}{N_s} e^{-w_{i_p}2\pi km/N_s}, \\
B(m,k) &= (-1)^k \sin\frac{w_{r_p}2\pi km}{N_s} e^{-w_{i_p}2\pi km/N_s}.
\end{aligned} \tag{12}$$

Similarly, real and imaginary parts of $x_{a_p}(n - w_{N,p}m/N)_q$ (Figure 3(b)) are obtained by using the constants:

$$\begin{aligned}
A_1(m,k) &= (-1)^k \cos\frac{w_{r_p}2\pi km}{N_s} e^{w_{i_p}2\pi km/N_s}, \\
B_1(m,k) &= (-1)^{k+1} \sin\frac{w_{r_p}2\pi km}{N_s} e^{w_{i_p}2\pi km/N_s},
\end{aligned} \tag{13}$$

instead of $A(m,k)$ and $B(m,k)$.

Note that the hardware components should provide satisfying precision even for large values of real exponential term $\exp(w_{i_p}2\pi mk/N_s)$ (for large m and k). The required precision depends on the number of input samples N_s and the distribution order N since $-N_s/2 \le m \le N_s/2 - 1$ and $-N_s/N \le k \le N_s/N$ hold. Thus, the miscalculation errors will be avoided if registers are designed to store the value $\log_2(e^{2\pi N_s/2})$. For example, if $N_s = 128$ and $N = 4$, the extended single precision IEEE-754 format should be used, while for $N_s = 256$, the extended double precision IEEE-754 format is required.

In the sequel, $x_{a_p}(n \pm w_{N,p}m/N)_q$ is used to calculate the concentration functions $c_{r_p}(n,m)_q$ and $c_{i_p}(n,m)_q$ (for the pth term in (4) and the qth signal component). The architecture for $c_{r_p}(n,m)_q$ and $c_{i_p}(n,m)_q$ is given in Figure 4. The realization is done according to

$$\begin{aligned}
\mathrm{Re}\left\{c_{r_p}(n,m)_q\right\} &= \mathrm{Re}\left\{e^{jw_{r_p}\,\mathrm{angle}((a+jb)/(c+jd))}\right\} \\
&= \cos\left(w_{r_p} \cdot \mathrm{atan}\left(\frac{bc-ad}{ac+bd}\right)\right), \\
\mathrm{Im}\left\{c_{r_p}(n,m)_q\right\} &= \mathrm{Im}\left\{e^{jw_{r_p}\,\mathrm{angle}((a+jb)/(c+jd))}\right\} \\
&= \sin\left(w_{r_p} \cdot \mathrm{atan}\left(\frac{bc-ad}{ac+bd}\right)\right), \\
\mathrm{Re}\left\{c_{i_p}(n,m)_q\right\} &= \mathrm{Re}\left\{e^{jw_{i_p}\ln|(c+jd)/(a+jb)|}\right\} \\
&= \cos\left(\frac{1}{2}w_{i_p}\left(\ln(c^2+d^2) - \ln(a^2+b^2)\right)\right), \\
\mathrm{Im}\left\{c_{i_p}(n,m)_q\right\} &= \mathrm{Im}\left\{e^{jw_{i_p}\ln|(c+jd)/(a+jb)|}\right\} \\
&= \sin\left(\frac{1}{2}w_{i_p}\left(\ln(c^2+d^2) - \ln(a^2+b^2)\right)\right),
\end{aligned} \tag{14}$$

where, to simplify the expressions, the following notations are used:

$$\begin{aligned}
a &= \mathrm{Re}\left\{x_{a_p}\left(n + w_{N,p}m/N\right)_q\right\}, \\
b &= \mathrm{Im}\left\{x_{a_p}\left(n + w_{N,p}m/N\right)_q\right\}, \\
c &= \mathrm{Re}\left\{x_{a_p}\left(n - w_{N,p}m/N\right)_q\right\}, \\
d &= \mathrm{Im}\left\{x_{a_p}\left(n - w_{N,p}m/N\right)_q\right\}.
\end{aligned} \tag{15}$$

The outputs of architecture in Figure 4 (defined by (14)), are further combined for all $p : p = 1, \ldots, N/2 - 1$, as

$$\begin{aligned}
c_r(n,m)_q &= \prod_{p=1}^{N/2-1} \left(\mathrm{Re}\left(c_{r_p}(n,m)_q\right) + j\ \mathrm{Im}\left(c_{r_p}(n,m)_q\right)\right), \\
c_i(n,m)_q &= \prod_{p=1}^{N/2-1} \left(\mathrm{Re}\left\{c_{i_p}(n,m)_q\right\} + j\ \mathrm{Im}\left\{c_{i_p}(n,m)_q\right\}\right).
\end{aligned} \tag{16}$$

The realizations of $c_r(n,m)_q$ and $c_i(n,m)_q$ are shown in Figures 5(a) and 5(b), respectively.

By taking all signal components, the resulting concentration functions $c_r(n,m)$ and $c_i(n,m)$ (Figure 6) are obtained:

$$\begin{aligned}
\mathrm{Re}\{c_r(n,m)\} &= \sum_{q=1}^{Q} \mathrm{Re}\left\{c_r(n,m)_q\right\}, \\
\mathrm{Im}\{c_r(n,m)\} &= \sum_{q=1}^{Q} \mathrm{Im}\left\{c_r(n,m)_q\right\},
\end{aligned}$$

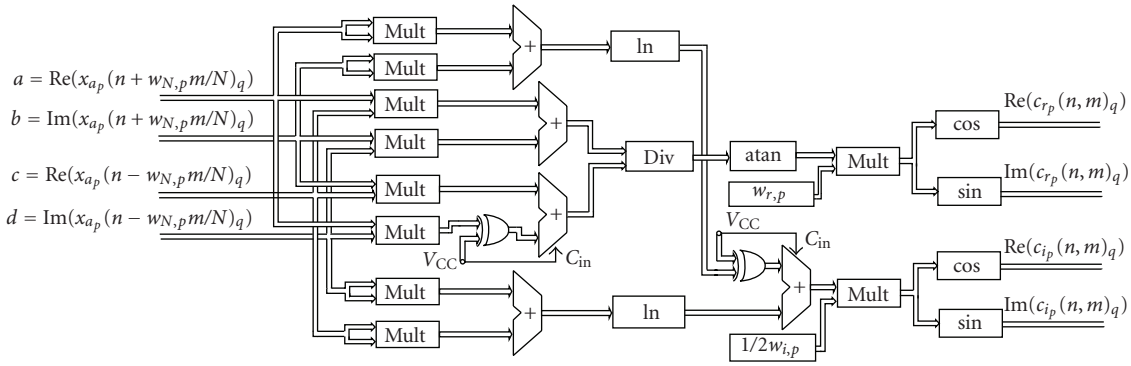
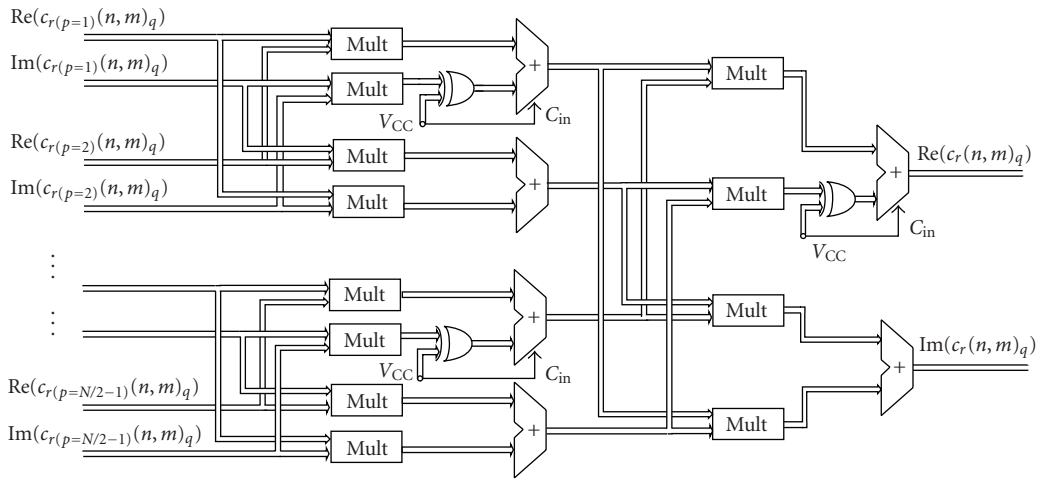
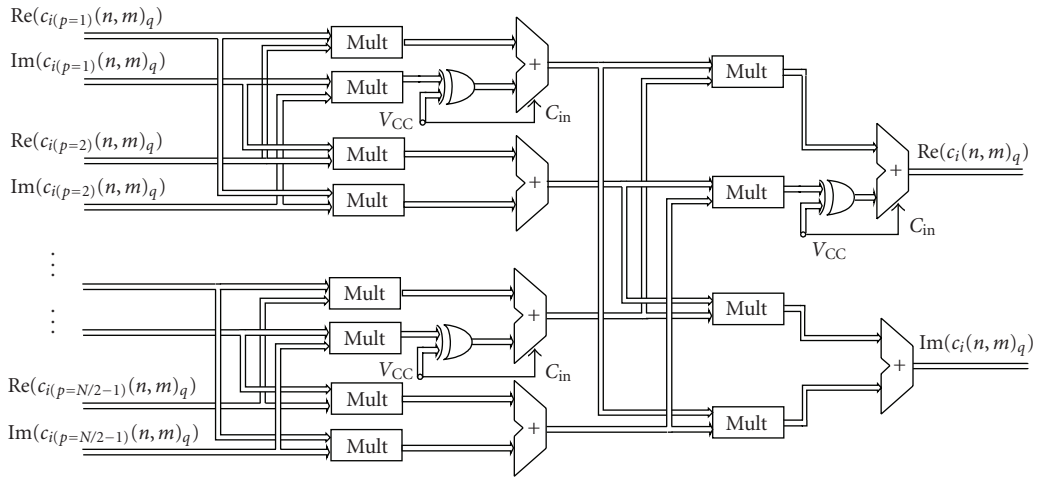


FIGURE 4: Architecture for realization of concentration functions $c_{r_p}(n,m)_q$ and $c_{i_p}(n,m)_q$.



(a)



(b)

FIGURE 5: Architecture for realization of concentration functions for the qth signal component: (a) $c_r(n,m)_q$ and (b) $c_i(n,m)_q$.

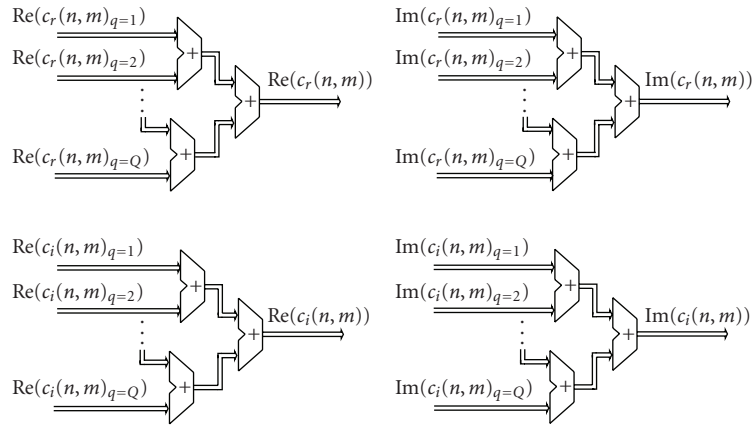


Figure 6: Architecture for resulting concentration functions $c_r(n,m)$ and $c_i(n,m)$.

$$\begin{aligned}\mathrm{Re}\{c_i(n,m)\} &= \sum_{q=1}^{Q} \mathrm{Re}\left\{c_i(n,m)_q\right\},\\ \mathrm{Im}\{c_i(n,m)\} &= \sum_{q=1}^{Q} \mathrm{Im}\left\{c_i(n,m)_q\right\}.\end{aligned} \tag{17}$$

3.2.3. BLOCK 3: Realization of Functions $C_r(n,k)$ and $C_i(n,k)$. The architecture in Figure 7 is used to obtain time-frequency domain functions: $C_r(n,k) = \mathrm{FFT}\{c_r(n,m)\}$, and $C_i(n,k) = \mathrm{FFT}\{c_i(n,m)\}$ (FFT denotes the fast Fourier transform). Since the real and imaginary parts are treated separately, the outputs of FFT circuits in Figure 7 are obtained in the form:

$$\begin{aligned}F_1(n,k) &= \mathrm{Re}\{\mathrm{FFT}(\mathrm{Re}\{c_r(n,m)\})\},\\ F_2(n,k) &= \mathrm{Im}\{\mathrm{FFT}(\mathrm{Re}\{c_r(n,m)\})\},\\ F_3(n,k) &= \mathrm{Re}\{\mathrm{FFT}(\mathrm{Im}\{c_r(n,m)\})\},\\ F_4(n,k) &= \mathrm{Im}\{\mathrm{FFT}(\mathrm{Im}\{c_r(n,m)\})\},\\ F_5(n,k) &= \mathrm{Re}\{\mathrm{FFT}(\mathrm{Re}\{c_i(n,m)\})\},\\ F_6(n,k) &= \mathrm{Im}\{\mathrm{FFT}(\mathrm{Re}\{c_i(n,m)\})\},\\ F_7(n,k) &= \mathrm{Re}\{\mathrm{FFT}(\mathrm{Im}\{c_i(n,m)\})\},\\ F_8(n,k) &= \mathrm{Im}\{\mathrm{FFT}(\mathrm{Im}\{c_i(n,m)\})\}.\end{aligned} \tag{18}$$

Therefore, the real and imaginary parts of functions $C_r(n,k)$ and $C_i(n,k)$ (the outputs of adders in Figure 7) follow as

$$\begin{aligned}\mathrm{Re}\{C_r(n,k)\} &= F_1(n,k) + F_3(n,k),\\ \mathrm{Im}\{C_r(n,k)\} &= F_2(n,k) + F_4(n,k),\\ \mathrm{Re}\{C_i(n,k)\} &= F_5(n,k) + F_7(n,k),\\ \mathrm{Im}\{C_i(n,k)\} &= F_6(n,k) + F_8(n,k).\end{aligned} \tag{19}$$

3.2.4. BLOCK 4: The Final Form of the Concentration Function in the Time-Frequency Domain. The concentration function $C(n,k)$ is obtained as a convolution of functions $C_r(n,k)$ and $C_i(n,k)$:

$$\begin{aligned}\mathrm{Re}\{C(n,k)\} &= \sum_{l=-L_d}^{L_d} \mathrm{Re}\{C_r(n,k+l)C_i(n,k-l)\},\\ \mathrm{Im}\{C(n,k)\} &= \sum_{l=-L_d}^{L_d} \mathrm{Im}\{C_r(n,k+l)C_i(n,k-l)\}.\end{aligned} \tag{20}$$

The architecture is shown in Figure 8(a) Note that, the realization of the product term $C_r(n,k+l)C_i(n,k-l)$, shown in Figure 8(b), is done according to

$$\begin{aligned}&\mathrm{Re}\{C_r(n,k+l)C_i(n,k-l)\}\\ &\quad= \mathrm{Re}\{C_r(n,k+l)\}\,\mathrm{Re}\{C_i(n,k-l)\}\\ &\qquad- \mathrm{Im}\{C_r(n,k+l)\}\,\mathrm{Im}\{C_i(n,k-l)\},\\ &\mathrm{Im}\{C_r(n,k+l)C_i(n,k-l)\}\\ &\quad= \mathrm{Re}\{C_r(n,k+l)\}\,\mathrm{Im}\{C_i(n,k-l)\}\\ &\qquad+ \mathrm{Im}\{C_r(n,k+l)\}\,\mathrm{Re}\{C_i(n,k-l)\}.\end{aligned} \tag{21}$$

In the final step, a convolution of $\mathrm{SM}(n,k)$ and $C(n,k)$ is performed within BLOCK 5 (Figure 1) and the channels of the Nth order complex-lag time-frequency distribution are obtained as

$$\begin{aligned}\mathrm{Re}\{\mathrm{MGCD}(n,k)\} &= \sum_{i=-L_d}^{L_d} \mathrm{Re}\{\mathrm{SM}(n,k+i)C(n,k-i)\},\\ \mathrm{Im}\{\mathrm{MGCD}(n,k)\} &= \sum_{i=-L_d}^{L_d} \mathrm{Im}\{\mathrm{SM}(n,k+i)C(n,k-i)\}.\end{aligned} \tag{22}$$

The corresponding architecture is given in Figure 9. Note that the convolution of $\mathrm{SM}(n,k)$ and $C(n,k)$ is realized in the same way as the convolution of $\mathrm{STFT}(n,k)$ and $\overline{\mathrm{STFT}}(n,k)$ within the SM.

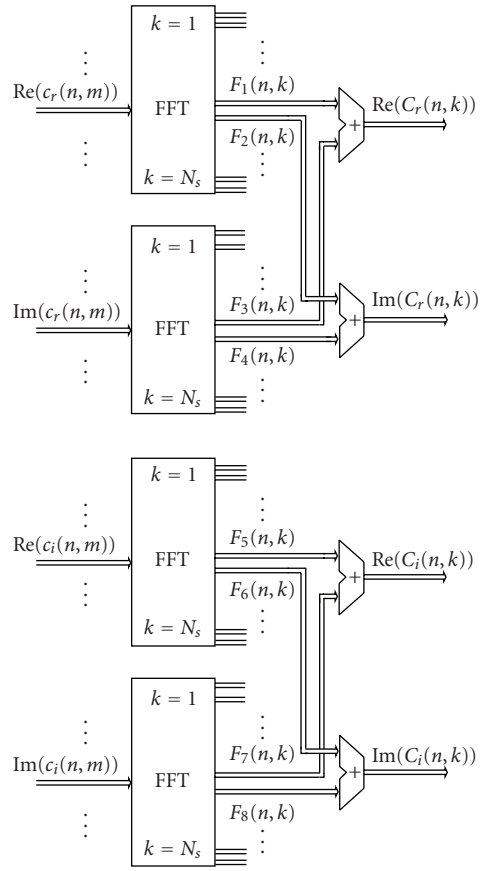


FIGURE 7: Architecture for realization of $C_r(n,k)$ and $C_i(n,k)$ functions.

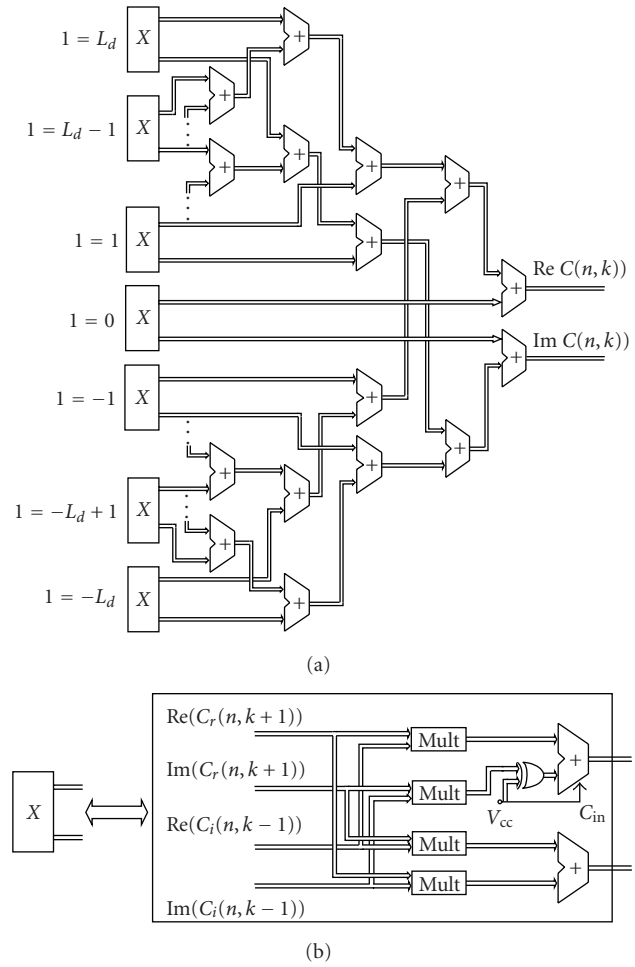


FIGURE 8: Architecture for realization of (a) $C(n,k)$ and (b) realization of product $C_r(n,k+l)C_i(n,k-l)$ for certain l.

4. Serial Architecture for Implementation of the Nth Order Complex-Lag Time-Frequency Distribution

As an alternative to the proposed system with parallel realization, a serial architecture for implementation of the N-th order complex-lag distribution is considered. The block scheme for serial architecture is given in Figure 10.

The STFT block is the same as in the parallel realization, while the serial architecture for the S-method (SM block) is given in [14]. The concentration function is obtained throughout the following blocks (Figure 10): BLOCK1, BLOCK2, BLOCK3, and BLOCK4. BLOCK1 (the separation of signal components) and BLOCK3 (the FFT calculation) are the same as in the parallel realization. Therefore, the main modifications with respect to the parallel realization are made within BLOCK2. It consists of Block21, Block22, Block23, and Block24, that will be briefly discussed in the sequel.

Block21. This block is used for the calculation of signal with complex-lag argument, given by (11). The realization is shown in Figure 11. The LUT (Look-Up Table) contains the constants: $A(m,k)$, $B(m,k)$, $A_1(m,k)$, and $B_1(m,k)$, defined by (12) and (13), respectively. Address 1 and Address 3 are used to provide synchronization between the STFT samples and the LUT elements. One summation term in (11) is obtained within one cycle of the clk6 clock, while the complete summation is performed within one cycle of clk1. The terms $\mathrm{Re}\{x_{ap}(n + w_{N,p}m/N)_q\}$, $\mathrm{Im}\{x_{ap}(n + w_{N,p}m/N)_q\}$, $\mathrm{Re}\{x_{ap}(n - w_{N,p}m/N)_q\}$, and $\mathrm{Im}\{x_{ap}(n - w_{N,p}m/N)_q\}$ are obtained within the four cycles of clk1. After each cycle of clk1, the RESET1 signal resets the cumulative adder ADD.

Block22. The outputs of Block21 are fed to the input of Block22 that is used to obtain concentration functions $c_{r_p}(n,m)_q$ and $c_{i_p}(n,m)_q$. The serial realization of these functions is the same as in the parallel architecture (Figure 4).

Block23. It calculates the resulting concentration functions $c_r(n,m)_q$ and $c_i(n,m)_q$ defined by (16). The realization is given in Figure 12. The Address 4 and Address 5 determine which pair of inputs should be multiplied within Mult circuit. One cycle of clk7 is set to the time interval required for one multiplication and one addition operation. After two cycles of clk7, RESET2 signal resets the clock ADD. The terms $\mathrm{Re}\{c_r(n,m)_q\}$, $\mathrm{Im}\{c_r(n,m)q\}$, $\mathrm{Re}\{c_i(n,m)_q\}$, and

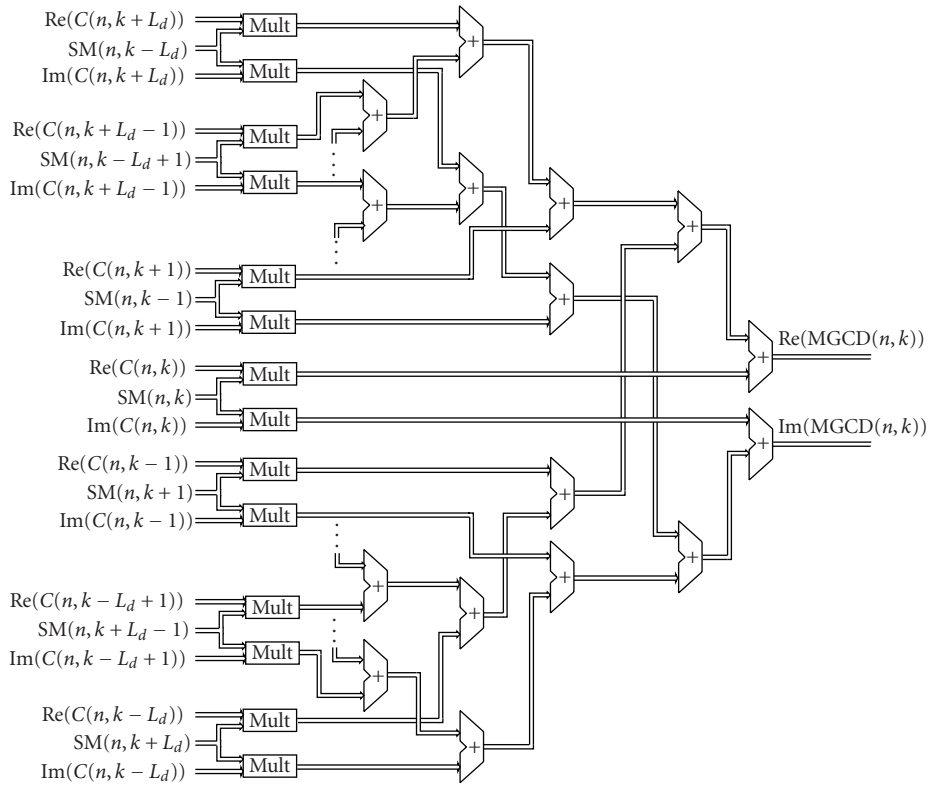


Figure 9: Architecture for realization of convolution between SM(n,k) and C(n,k).

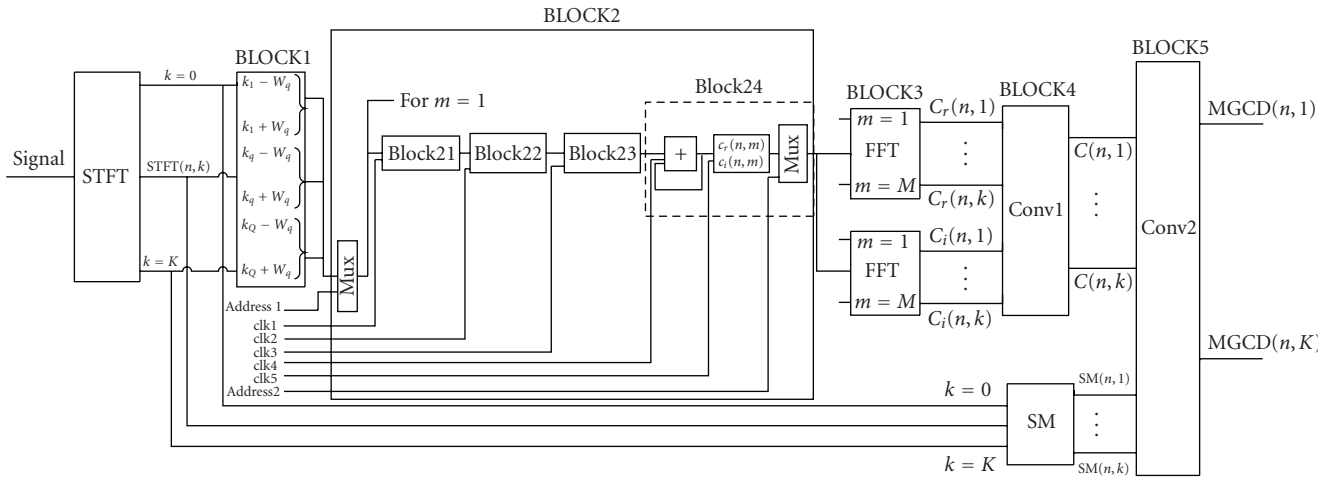


Figure 10: Serial architecture for realization of generalized complex-lag distribution for multicomponent signals.

Im$\{c_i(n,m)_q\}$ are obtained after 4(N/2-1) cycles. The cumulative multiplier Mult1 should be reset after each N/2-1 cycles of clk8.

Block24. The final concentration functions $c_r(n,m)$ and $c_i(n,m)$ are obtained at the output of this block. One cycle of the clk4 is set to the time interval that is necessary for the calculations within the three previous blocks. Address 2 determines the order of inputs in BLOCK3, Figure 10. The concentration functions $C_r(n,k)$ and $C_i(n,k)$ are obtained at the output of BLOCK3.

BLOCK4 is used to perform the calculation of concentration function $C(n,k)$ defined by (20). The realization of this block is given in Figure 13. One term in summation is calculated within one cycle of the clk9, while complete sum is obtained after $2L_d + 1$ cycle. The clock cycles and reset signals of this circuit are similar as in Block23. The resulting complex-lag distribution is obtained at the output of BLOCK5 in Figure 10. The realization of this block is similar to the realization of BLOCK4. The only difference is that only one cumulative adder is required.

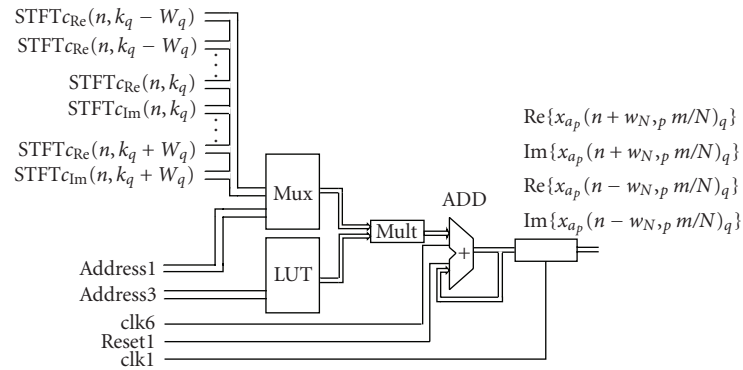


Figure 11: Serial architecture for producing of signal with complex-lag argument (Block21).

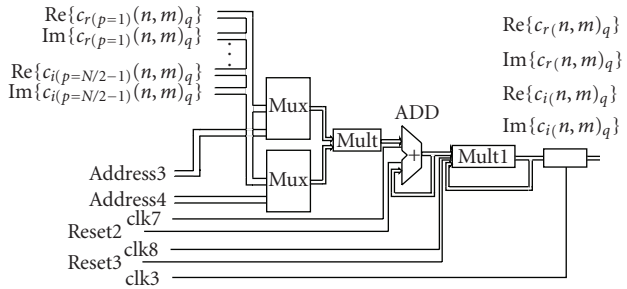


Figure 12: Serial Architecture for realization of concentration functions $c_r(n, m)_q$ and $c_i(n, m)_q$ (Block23).

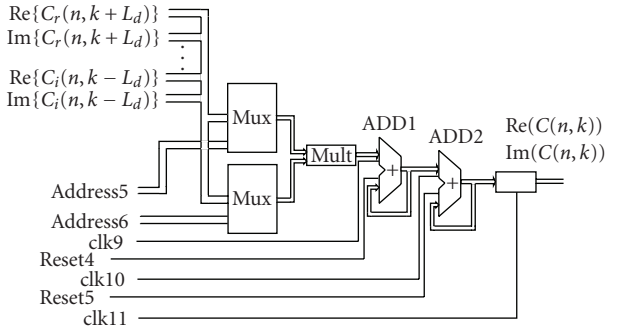


Figure 13: Serial architecture for realization of $C(n, k)$ (BLOCK 4).

Table 1: Number of circuits for parallel realization.

	Adders	Multipliers	Other circuits
Figure 2	$2(L_d + 2)$	$2(L_d + 3)$	
Figure 3	$8W_q + 2$	$8W_q + 4$	
Figure 4	$5(N/2 - 1)$	$10(N/2 - 1)$	2ln, divide, atan, cos, sin
Figure 5	$N - 4$	$4(N/2 - 2)$	
Figure 6	$4(Q - 1)$		
Figure 7	$4(2 + \log_2 N_s)$	$4\log_2 N_s$	
Figure 8	$8L_d + 2$	$8L_d + 4$	
Figure 9	$4L_d$	$2L_d + 2$	

5. Analysis of the Proposed System

In the sequel, some practical issues related to the realization of the proposed hardware realizations are addressed. The proposed system (either with parallel or serial architecture) can be implemented by using the hardware components with floating point format to provide satisfying precision for the calculation of signal with complex-lag argument. However, the floating point adders and multipliers introduce the output latency of few clock cycles. Thus, in order to decrease the output latency, the number of hardware components with floating point format should be reduced. The large values that require high precision appear only in the realization of concentration functions $c_{r_p}(n, m)_q$ and $c_{i_p}(n, m)_q$ (Figures 3 and 4 in parallel realization, Figure 11 in serial realization). The remaining part of the system can be realized by using the fixed point format. Note that starting from the output of the system shown in Figure 4 (the outputs of cos and sin circuits), all values will be in the range $[-1, 1]$, and the fixed point notation could be used. Also, by normalization of input signal, the STFT and the SM can be calculated by using the fixed point format.

The floating point multipliers increment the exponent by 1, which should be corrected at the output. The fixed point multipliers result in a two sign-bit. Thus, to correct the result, the product has to be shifted left by one bit. This shifter can be included as a part of multiplier.

The total number of circuits needed for one channel of the proposed system for parallel realization is given in Table 1. The longest path is given in Table 2. It connects the register storing STFT$(n - 1, k \pm W_q)$ with the output MGCD(n, k). The length of this path determines the fastest sampling rate.

The number of required circuits and longest path for serial realization are given in Tables 3 and 4, respectively.

In order to compare the parallel and serial realizations the following values are considered: $N_s = 128$, $N = 6$, $W_q = 2$, $L_d = 2$, and $Q = 2$. The number of hardware components and the latency for parallel and serial realizations are given in Table 5.

Note that the number of components required for serial realization is reduced with respect to parallel realization. However, the latency in serial realization is significantly increased. Therefore, in order to provide higher processing

Table 2: The longest path of parallel realization.

	Adders	Multipliers	Other circuits
Figure 2	$L_d + 3$	2	
Figure 3	$2W_q$	1	
Figure 4	1	2	(cos or sin) and ((div+atan) or (ln+add))
Figure 5	$\lceil \log_2(N/2-1) \rceil$	$\lceil \log_2(N/2-1) \rceil$	
Figure 6	$\lceil \log_2 Q \rceil$		
Figure 7	$1 + \log_2 N_s$	$\log_2 N_s$	
Figure 8	$L_d + 2$	1	
Figure 9	$L_d + 1$	1	

Table 3: Number of circuits for serial realization.

	Adders	Multipliers	Other circuits
STFT	4	4	
SM	1	1	Mux
Block21	1	1	Mux, LUT
Block22	5	10	2ln, divide, atan, cos, sin
Block23	1	2	2 Mux
Block24	1		Mux
BLOCK3	$4(2 + \log_2 Ns)$	$4 \log_2 N_s$	
BLOCK4	4	4	Mux
BLOCK5	2	2	Mux

Table 4: The longest path of serial realization.

	Adders	Multipliers	Other circuits
STFT	2	1	
SM	$2L_d$+1	L_d+1	
Block21	$4(2Wq+1)$	$4(2Wq+1)$	
Block22	$N/2$-1	$2(N/2$-1)	((cos or sin) and ((div+atan) or (ln+add)))
Block23	$(N/2$-1)	$2(N/2$-1)	
Block24	Q	Q	
BLOCK3	$\log_2 N_s$	$\log_2 N_s$	
BLOCK4	$2(2L_d+1)$	$2L_d$+1	
BLOCK5	$2L_d$+1	$2L_d$+1	

Table 5: Comparison between parallel and serial realization.

	Parallel realization		Serial realization	
	Adders	Multipliers	Adders	Multipliers
No. of circuits	190	216	55	52
Latency	25	15	189	341

speed, we will consider parallel realization for the FPGA implementation.

6. FPGA Implementation of the Proposed Parallel Architecture

The proposed architecture can be implemented by using various hardware devices. As one of the possible choices, the digital signal processor could be used. However, it is not suitable for real-time processing, especially at very high speeds. Therefore, for high-speed processing, one might consider the ASIC implementation (application specific integrated circuit) or the FPGA implementation. Both of them allow a high degree of parallelism, as well. ASIC implementation can provide lower power consumption, design size optimization, and design flexibility that enables speed optimization. However, long production time and significant costs do not recommend ASIC device for prototype development. The main advantages of the FPGA implementation are reconfigurability, lower producing time and costs requirements, inbuilt special hardware such as RAM, and so forth. Thus, we use the FPGA to develop the prototype that, after testing and verifying the results, can be implemented on an ASIC. The FPGA chips from Altera [17] and the Quartus II v8.0 software are used in the proposed implementation.

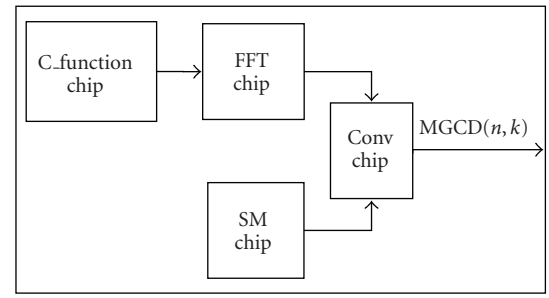


Figure 14: A block scheme for the FPGA implementation.

Table 6: Characteristics for the chip in Figure 16.

EP3C40F780C6	No. of pins	Logic elements	Speed	Power consumption
Available	535	39600	500 MHz	1.2 V
Utilized	474 (88%)	8284 (21%)	91 MHz	1.2 V

Table 7: Characteristics for the chip in Figure 17.

EP3C16F484C6	No. of pins	Logic elements	Speed	Power consumption
Available	347	15408	500 MHz	1.2 V
Utilized	220 (63%)	6836 (44%)	158 MHz	1.2 V

The fourth-order ($N = 4$) complex-lag distribution is considered for FPGA implementation. A simplified block scheme of the system is shown in Figure 14. The FPGA implementation of the S-method (SM Chip), including the chip parameters and performance, has been provided in [14]. Thus, we focus on designing the chip for the concentration function $c(n, m)$ (C_function Chip). The output of this chip is passed to the input of the FFT Chip (that produce $C(n, k)$). Note that a solution for the FFT chip by using the FFT MegaCore function has recently been proposed by Altera [18]. For device EP3SE50F780C2 (Stratix III family) and the 16-bit fixed point format, this FFT chip requires 4290 ALUTs, 5116 memory bits, 20 DSP 18×18 multipliers, and

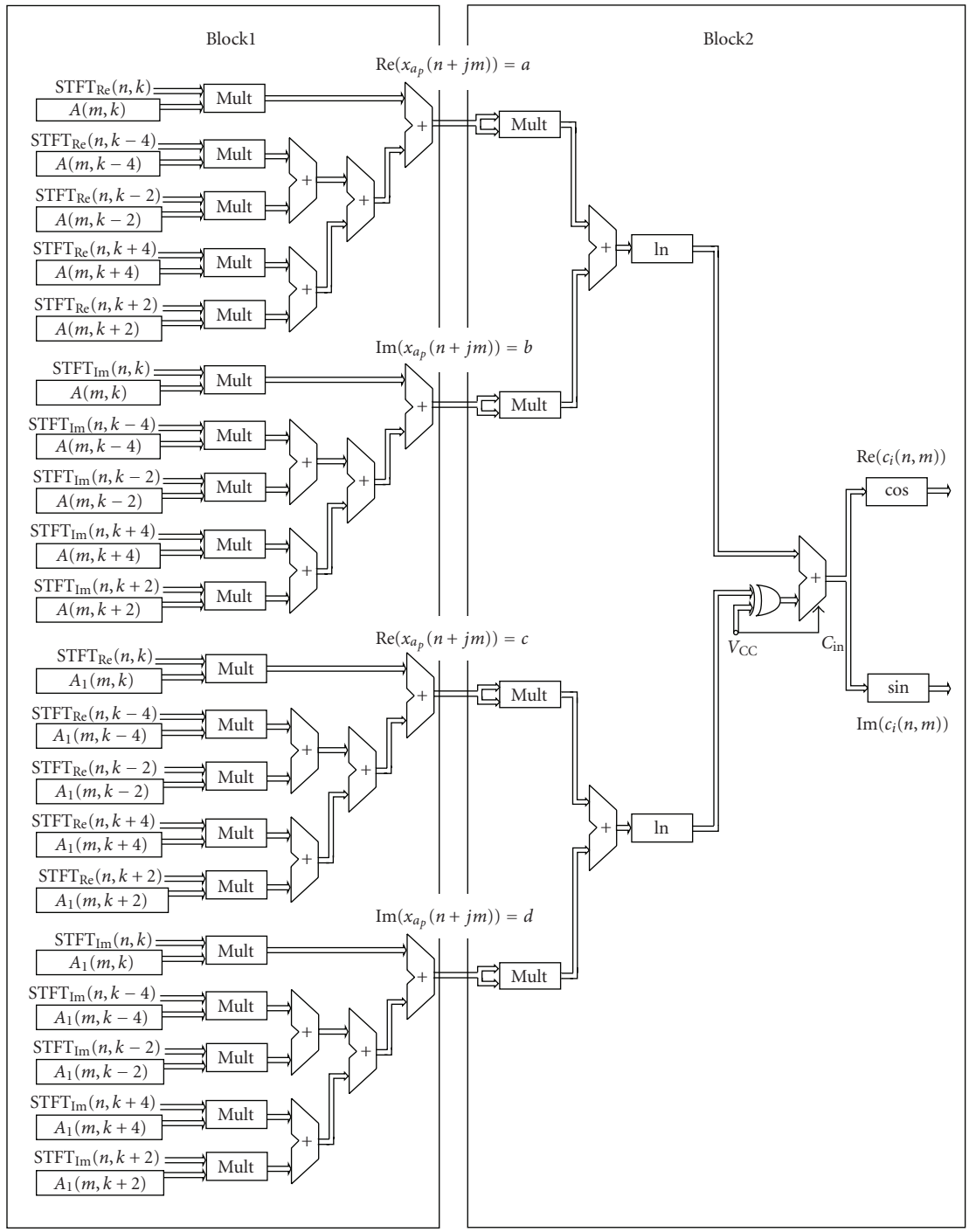


Figure 15: Architecture for concentration function calculation.

transform time 0.37 microseconds. The outputs from the SM chip and the FFT chip are convolved in the final step, where the convolution circuit (Conv Chip) is realized in the same way as the SM chip.

Realization of the C_function Chip. In the case $N = 4$, $w_{N,p} = w_r + jw_i = j$ holds. Consequently, the concentration function is obtained as $c(n,m) = c_i(n,m)$, since $c_r(n,m) = 1$. The calculation of signal with complex-lag argument is reduced with respect to (11), since $B(m) = 0$, and the corresponding architecture is shown in Figure 15 (the Block 1). The remaining components for the realization are given in Block 2, Figure 15. The schematic diagram for FPGA realization of signal $\mathrm{Re}\{x_{a_p(n+jm)}\} = a$ is given in Figure 16. In order to avoid miscalculations of the analytic extension, the extended single precision floating point arithmetic is used (the number of samples is $N_s = 128$). The input pins

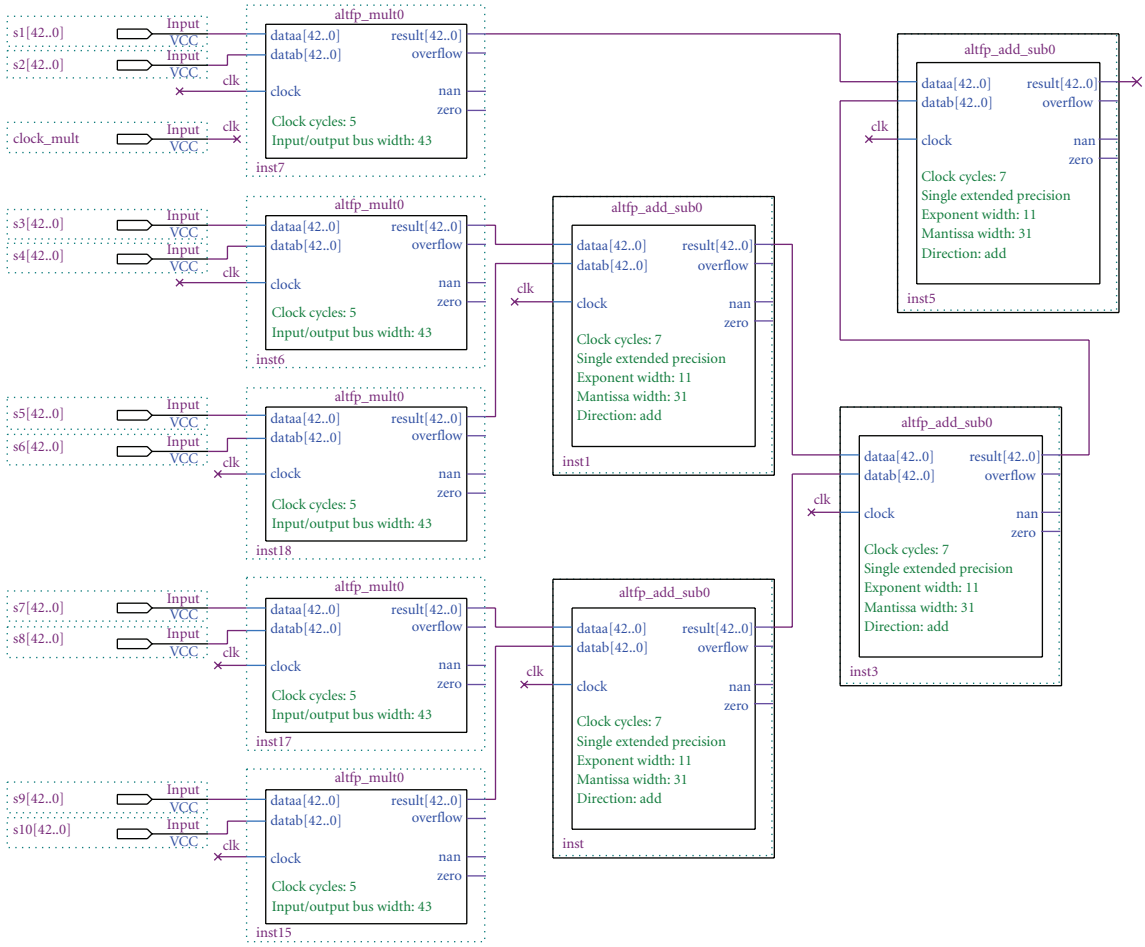


Figure 16: FPGA realization of signal with complex-lag argument.

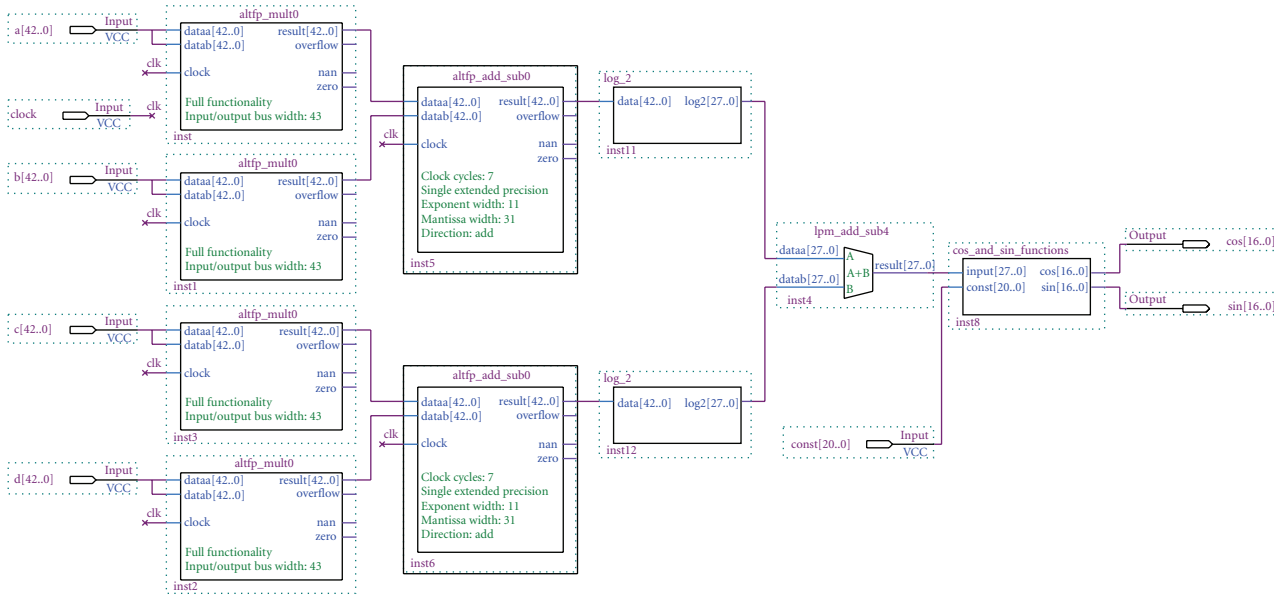


Figure 17: FPGA realization of concentration function.

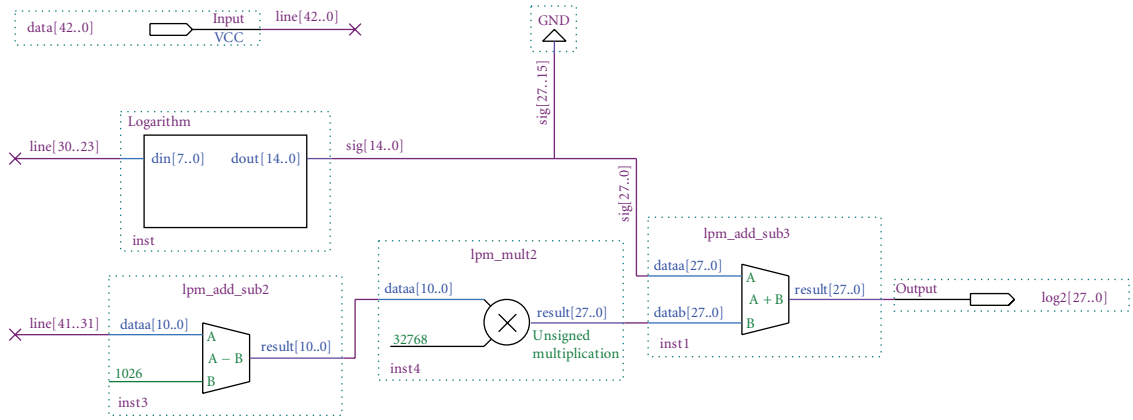


Figure 18: FPGA realization of logarithm function (log_2 function).

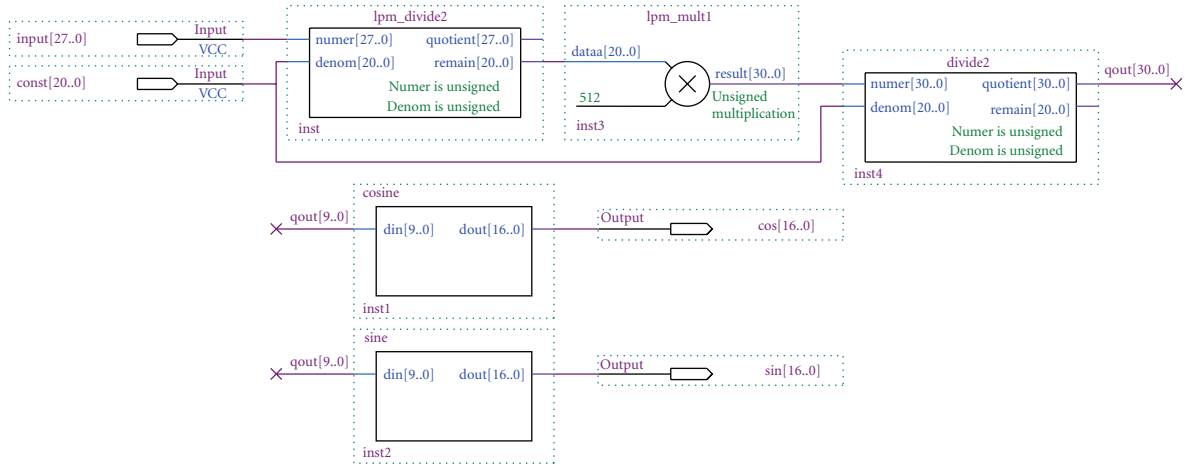


Figure 19: FPGA realization of cosine and sine function (cos_and_sin function).

s1[42...0], s3[42...0],..., s9[42...0] represent STFT($m, k-4$), STFT($m, k-2$),..., STFT(m, k), respectively. The 43-bit values $A(m, k-4)$, $A(m, k-2)$, ... $A(m, k)$ are fed to the input pins s2[42...0], s4[42...0],..., s10[42...0], respectively.

The floating point multiplier altfp_mult0 has the output latency of five clock cycles, while the output latency of floating point adder altfp_add_sub0 is seven clock cycles. The output latency is determined by the number of summation terms in (11). Here, one multiplier and W_q+1 adder are used. Note that the multiplier altfp_mult0 produces the result that is multiplied by two, which will be corrected in the circuit for cosine function calculation.This architecture is realized in the device EP3C40F780C6 from Cyclone III family fabricated in DDR2-SDRAM technology. The available chip characteristic and utilized resources are given in Table 6.

The schematic diagram for FPGA realization of the second part (Block 2 from Figure 15) is given in Figure 17. The input pins are a[42...0], b[42...0], c[42...0], and d[42...0] representing the outputs of circuits that calculates the signal with complex-lag argument. Note that input pin const [20...0] represents a correction term used in the calculation of cosine function (more details are given in the appendix). The architecture in Figure 17 is realized in the device EP3C16F484C6 from Cyclone III family fabricated in DDR2-SDRAM technology. The available chip characteristic and utilized resources are given in Table 7.

The circuits that calculate natural logarithm and cosine and sine functions are not implemented in the Quartus II v8.0 software. Thus, they could be calculated by using polynomial approximations, CORDIC (COordiante Rotation DIgital Computer) algorithm or LUT. The approach that uses polynomial approximations requires floating point arithmetic and iterative calculations. This method is computationally very extensive, slow, and it is not suitable for high-speed processing. CORDIC is a recursive algorithm that uses small number of circuits to provide calculation of trigonometric, logarithmic, and exponential functions [18–21]. If very high precision is required, the CORDIC approach will be more suitable and then the LUT. It has been shown that for the precision up to 16-bits, the LUT approach provides higher processing speeds [21]. The test we performed shows that 16 bit precision is sufficient for the calculation of the logarithmic and trigonometric functions and higher precision does not provide further improvement of results. Thus, for the calculation of logarithmic and trigonometric functions we use the LUT approach. Simple

solutions for these functions (log_2 circuit and cos_and_sin function circuit in Figure 17) are given in the appendix.

Finally, we have compared the throughput rate and latency for software simulation and proposed hardware realization. The software simulation is performed by using MATLAB 7 running on Pentium IV with 3.2 GHz and 1 GB RAM. The throughput rate obtained in the software simulation is 4.6 Mbs, while the latency is 90 microseconds. The proposed hardware realization provides throughput rate 6.9 Gbs and latency of 180 nanoseconds. Therefore, as expected, the proposed hardware realization provides significant improvement for throughput rate and latency compared to the software simulation.

7. Conclusion

The proposed hardware provides an efficient implementation of the Nth order complex-lag time-frequency distribution for multicomponent signals. This flexible system is realized in parallel and serial configurations and combines fixed and floating point arithmetic. It provides satisfactory calculation precision and solves the problem of errors that could appear in numerical realizations of distributions with complex-lag argument. Furthermore, the FPGA implementation of the proposed parallel hardware solution is provided, including the parameters of FPGAs chips.

Appendix

Natural Logarithm. Since we deal with the binary arithmetic, the natural logarithm of number X can be written in terms of the logarithm with base 2: $\ln X = \log_2 X/\log_2 e$. Let us consider the floating-point number in the form $X = x_1 2^{x_2}$, where x_1 and x_2 are mantissa and exponent, respectively. Thus, $\log_2 x_1 2^{x_2} = x_2 + \log_2 x_1$ holds. The schematic diagram for the calculation of logarithm with base 2 is shown in Figure 18. The exponent x_2 has been incremented for the value 1026 within the preceding circuits (1023 comes from the extended single precision floating point format, while 3 is introduced by floating point multipliers). Thus, in order to correct the result, the exponent is reduced for the same value within the lpm_add_sub2 circuit in Figure 18. The calculation of $\log_2 x_1$ is performed by using the look-up table (LUT) that contains 255 values (logarithm circuit in Figure 18). The first 8 bits of mantissa are used as an input of LUT. In order to obtain satisfying precision, it is created as $\mathrm{LUT}(x_1) = \mathrm{round}(2^{15}\log_2 x_1)$. Consequently, we have

$$\log_2 x_1 2^{x_2} = x_2 + \frac{\mathrm{LUT}(x_1)}{2^{15}} = \frac{2^{15}x_2 + \mathrm{LUT}(x_1)}{2^{15}}. \quad (A.1)$$

Note that $2^{15}x_2$ is obtained at the output of circuit lpm_mult2 in Figure 17. Thus, the final output log2 [27 . . . 0] has the value $(2^{15} \cdot x_2 + \mathrm{LUT}(x_1))$.

Cosine and sine functions. The input value in the cos_and_sin_function circuit, given in Figure 19, is first corrected by scaling with constant value $1/(2 \cdot 2^{15}\log_2 e)$ (const[20..0] in Figures 17 and 19) in the circuit lpm_divide2. Note that 1/2 results from (14), while $1/(2^{15}\log_2 e)$ remains from the calculation of natural logarithm. The input of this circuit is additionally divided by the period of cosine function 2π. Then the remainder (remain [20..0] in Figure 19) is used as an input of LUT that provides the values of cosine and sine functions. Namely, in order to provide satisfying precision, the remainder is quantized to 512 values. The ten least significant bits, obtained by the quantization, are used as an input in the LUT that provides 16-bit values for cosine and sine functions.

References

[1] L. Cohen, "Time-frequency distributions—a review," *Proceedings of the IEEE*, vol. 77, no. 7, pp. 941–981, 1989.

[2] P. Loughlin, Ed., "Special issue on time-frequency analysis," *Proceedings of the IEEE*, vol. 84, no. 9, 1996.

[3] B. Boashash, *Time-Frequency Signal Analysis*, Elsevier, Oxford, UK, 2003.

[4] LJ. Stanković, "A method for time-frequency analysis," *IEEE Transactions on Signal Processing*, vol. 42, no. 1, pp. 225–229, 1994.

[5] LJ. Stanković, "A method for improved distribution concentration in the time-frequency analysis of multicomponent signals using the L-Wigner distribution," *IEEE Transactions on Signal Processing*, vol. 43, no. 5, pp. 1262–1268, 1995.

[6] B. Boashash and P. O'Shea, "Polynomial Wigner-Ville distributions and their relationship to time-varying higher order spectra," *IEEE Transactions on Signal Processing*, vol. 42, no. 1, pp. 216–220, 1994.

[7] B. Barkat, "Instantaneous frequency estimation of nonlinear frequency-modulated signals in the presence of multiplicative and additive noise," *IEEE Transactions on Signal Processing*, vol. 49, no. 10, pp. 2214–2222, 2001.

[8] S. Stanković and LJ. Stanković, "Introducing time-frequency distribution with a "complex-time" argument," *Electronics Letters*, vol. 32, no. 14, pp. 1265–1267, 1996.

[9] LJ. Stanković, "Time-frequency distributions with complex argument," *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 475–486, 2002.

[10] M. Morelande, B. Senadji, and B. Boashash, "Complex-lag polynomial Wigner-Ville distribution," in *Proceedings of IEEE Region 10 Annual International Conference on Speech and Image Technologies for Computing and Telecommunications (TENCON '97)*, vol. 1, pp. 43–46, December 1997.

[11] G. Viswanath and T. V. Sreenivas, "IF estimation using higher order TFRs," *Signal Processing*, vol. 82, no. 2, pp. 127–132, 2002.

[12] C. Cornu, S. Stanković, C. Ioana, A. Quinquis, and LJ. Stanković, "Generalized representation of phase derivatives for regular signals," *IEEE Transactions on Signal Processing*, vol. 55, no. 10, pp. 4831–4838, 2007.

[13] S. Stanković, N. Žarić, I. Orović, and C. Ioana, "General form of time-frequency distribution with complex-lag argument," *Electronics Letters*, vol. 44, no. 11, pp. 699–701, 2008.

[14] S. Stanković, LJ. Stanković, V. Ivanović, and R. Stojanović, "An architecture for the VLSI design of systems for time-frequency analysis and time-varying filtering," *Annals of Telecommunications*, vol. 57, no. 9-10, pp. 974–995, 2002.

[15] A. Papoulis, *Signal Analysis*, McGraw-Hill, New York, NY, USA, 1997.

[16] M. Unser, "Recursion in short-time signal analysis," *Signal Processing*, vol. 5, no. 3, pp. 229–240, 1983.

[17] *Cyclone III, Device Datasheet: DC and Switching Characteristic*, Altera Corporation, 2007.
[18] *FFT MegaCore Function*, Altera Corporation, 2008.
[19] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '98)*, pp. 191–200, Monterey, Calif, USA, February 1998.
[20] J. Hormigo, J. Villalba, and E. L. Zapata, "Interval sine and cosine functions computation based on variable-precision CORDIC algorithm," in *Proceedings of the 14th Symposium on Computer Arithmetic*, pp. 186–193, April 1999.
[21] T. Vladimirova and H. Tiggeler, "FPGA implementation of sine and cosine generators using the CORDIC algorithm," in *Proceedings of the MAPLD International Conference*, Washington, DC, USA, September 2006.